Similarity-Based Supervisory Control of Discrete Event Systems

Yongzhi Cao and Mingsheng Ying

Abstract

Due to the appearance of uncontrollable events in discrete event systems, one may wish to replace the behavior leading to the uncontrollability of pre-specified language by some quite similar one. To capture this similarity, we introduce metric to traditional supervisory control theory and generalize the concept of original controllability to λ-controllability, where λ indicates the similarity degree of two languages. A necessary and sufficient condition for a language to be λ-controllable is provided. We then examine some properties of λ-controllable languages and present an approach to optimizing a realization.

Index Terms

Discrete event systems, supervisory control, controllability, metric space, Pareto optimality.

I. Introduction

Supervisory control theory (SCT) initiated by Ramadge and Wonham [16] and subsequently extended by other researchers (see, for example, [2], [17] and the bibliographies therein) provides a systematic approach to controlling discrete event systems (DES). The behavior of a DES is represented by a language over the set of events, and in the paradigm of standard SCT, Ramadge and Wonham [16] have formulated supervisory control problem by two languages that correspond

This work was supported by the National Foundation of Natural Sciences of China (Grant No: 60496321, 60321002, 60273003, 60505011), the Key Grant Project of Chinese Ministry of Education (Grant No: 10403), and the Chinese National Key Foundation Research & Development Plan (2004CB318108).

The authors are with the State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: caoyz@mail.tsinghua.edu.cn, yingmsh@mail.tsinghua.edu.cn).

to minimal acceptable behavior and legal behavior, respectively. In this formulation, both general and nonblocking solutions are well discussed.

Because of some practical requirements of control engineering, the standard SCT has been extended from the aspect of control objective. It has been observed by Lafortune and Chen [7] that the control objective of requiring nonblocking solutions is too conservative in some cases, and thus they have developed the supervisory control problem with blocking in [3], [7]. Subsequently, Lafortune and Lin [8], [9] formulated and solved a more general supervisory control problem whose control objective is given by “desired” behavior and “tolerated” behavior. The motivation behind this is that one can achieve more desired behavior by tolerating some behavior that will exceed the ideal desired one. Using probability to precisely specify what is tolerable was first presented by Lin in [11]. This work was further developed in probabilistic DES [10]. The research mentioned above shows that to achieve more desired behavior, sometimes it is worth tolerating undesirable behavior especially in some systems whose constraints are not rigid.

Tolerable behavior, which depends on different practical systems, gives rise to different supervisory control problems. In this paper, we are interested in the supervisory control problem in which one can accept some behavior quite similar to desired one. This is motivated by the fact that some similar behavior often occurs in some DES and one may wish to tolerate similar behavior when the ideal desired one is not feasible. For example, assume that in a common computer system, the jobs completed by CPU (central processing unit) will request access to peripheral devices consisting of one printer and one disk. It seems reasonable to expect that if the default device is busy or wrong, the jobs will give access to the other device.

In order to capture the similarity of behavior, we first suppose that the event set of a DES is equipped with a metric d. This hypothesis is not too constrained since any nonempty set can be endowed with at least the discrete metric. The metric d indicates the similarity of events. Based upon this metric, we then construct a distance function $\tilde{d}$ for all pairs of event strings by using so-called Baire metric. Finally, the Hausdorff metric $\tilde{d}_H$ induced by $\tilde{d}$ can serve as a similarity measure on the set of languages. The less the value of $\tilde{d}_H$, the more similar the two languages. With this similarity measure, we propose the concept of λ-controllability, where λ stands for similarity index. More explicitly, we say that a language K is λ*-controllable* if there exists a controllable language $\widetilde{K}$ satisfying that $\tilde{d}_H(\widetilde{K}, K) \leq \lambda$. Such a $\widetilde{K}$ is called a *realization* of K. Clearly, each controllable language in the sense of SCT is λ-controllable, and moreover,

0-controllability coincides with original controllability. Hence, the notion of λ-controllability is a generalization of the original controllability in SCT. In some applications, the specifications offered by users may be relaxed. If a specification is not controllable and some dissimilarities between events can be tolerated, then we can turn our attention to finding a similar one by using λ-controllability, which increases the intelligence of standard supervisory control.

In our setting, we still use the traditional supervisor to control the system; the control objective which is different from the aforementioned ones is, however, to find a realization of the pre-specified desired language. In other words, the control objective here is to achieve certain behavior similar to the desired one. Taking similarity of elements into account and using metric to describe the similarity are widely recognized in some fields of Computer Science such as metric semantics, process calculus, and pattern recognition (see, for example, [4], [21], [20]). In the earlier work [15], a distance function defined in [5] is also used to characterize the infinite or sequential behavior of DES, and moreover, a generalized notion of controllability for ω-languages is introduced. Such a notion essentially depends on the prefix of ω-language under consideration, and thus it cannot serve our purpose of similarity-based supervisory control. Recently, a signed real measure for sublanguages of regular languages has been formulated and studied in [18], [19]. The measure which is different from our similarity measure only serves as an evaluation of supervisors. Perhaps there is a deep connection between them, and this is an interesting problem for the future study. Related to the metric for events, in Petri nets the synchronic distance between transitions has been introduced by Petri [14] to describe the degree of mutual dependence between events in a condition/event system (see [13] and the bibliographies therein for further information on synchronic distances).

The purpose of this paper is to introduce the idea of similarity-based supervisory control, and we only concentrate on some basic aspects of λ-controllability. We first examine some algebraic properties of λ-controllable languages, and then present a necessary and sufficient condition for a language to be λ-controllable. An algorithm for determining whether or not a finite language is λ-controllable is also provided. Further, we show that the supremal λ-controllable sublanguage of a given language exists, and discuss some of its properties. Finally, for a given λ-controllable language K, we turn our attention to finding a Pareto optimal realization $\widetilde{K}$ in the sense that it is impossible to enlarge the common behavior $\widetilde{K}\cap K$ and simultaneously reduce the different behavior $\widetilde{K}\backslash K$.

The rest of the paper is organized as follows. In Section II, we review some basics of SCT and metric space. In Section III, we introduce the concept of λ-controllability, discuss some properties of λ-controllable languages, and present a necessary and sufficient condition for a language to be λ-controllable. The supremal λ-controllable sublanguage is addressed in this section as well. Section IV is devoted to deriving a Pareto optimal realization from an arbitrary realization. We provide an illustrative example in Section V and conclude the paper in Section VI. Due to the lack of space, some proofs and technical details are omitted[1].

II. Preliminaries

Let E denote the finite set of events, and E^* denote the set of all finite sequences of events, or stings, in E, including the empty string ϵ. The length of a string ω is denoted by $l(\omega)$, and the prefix closure of a language L is denoted by $\overline{L}$.

The DES to be controlled is modelled by a deterministic automaton: $G = (Q, E, \delta, q_0)$, where Q is a set of states with the initial state q_0, E is a set of events, and $\delta : Q \times E \to Q$ is a (partial) transition function. The function δ is extended to $\delta : Q \times E^* \to Q$ in the obvious way. The behavior of a DES is modelled as a prefix closed language $L(G) = \{s \in E^* : \delta(q_0, s) \text{ is defined}\}$.

The supervisory control theory partitions the event set into two disjoint sets of controllable and uncontrollable events, E_c and E_{uc}, respectively. A supervisor is a map $S : L(G) \to 2^E$ such that $S(s) \supseteq E_{uc}$ for any string $s \in L(G)$. The language generated by the controlled system is denoted by $L(S/G)$. Following [16], a language $K \subseteq L(G)$ is said to be *controllable* (with respect to $L(G)$ and E_{uc}) if $\overline{K}E_{uc} \cap L(G) \subseteq \overline{K}$. It has been shown in [16] that a given nonempty language $K \subseteq L(G)$ is controllable if and only if there exists a supervisor S such that $L(S/G) = \overline{K}$.

For any language K, there exist the supremal controllable sublanguage [16] and the infimal prefix closed and controllable superlanguage [7] of K, denoted by $K^{\uparrow}$ and $K^{\downarrow}$, respectively. For more details about the theory of DES, we refer the reader to, for example, [2].

Let us collect some basic notions on metric space.

Definition 1: A *(1-bounded) metric space* is a pair (X, d) consisting of a nonempty set X and a function $d : X \times X \longrightarrow [0, 1]$ which satisfies the following conditions:

(M1) $d(x, y) = 0$ if and only if $x = y$,

[1]The details can be found at http://arxiv.org/abs/cs.DM/0410031.

(M2) $d(x,y) = d(y,x)$ for all $x, y \in X$, and

(M3) $d(x,z) \leq d(x,y) + d(y,z)$ for all $x, y, z \in X$.

The distance $d(x,y)$ measures the similarity between x and y. The less the distance, the more similar the two elements. To simplify notation, sometimes we write X instead of (X,d). Recall that if (X,d) is a metric space and $M \subseteq X$, then $(M, d|_{M\times M})$ is also a metric space, where $d|_{M\times M}$ is the restriction of d to M.

Let (X,d) be a metric space, $x_0 \in X$, and $\lambda \in [0,1]$. The set $B(x_0,\lambda) = \{x \in X : d(x_0,x) \leq \lambda\}$ is called the *λ-ball about* x_0; for a subset A of X, by the *λ-ball about* A we mean that the set $B(A,\lambda) = \cup_{x\in A} B(x,\lambda)$. We extend d to a pair x, A, where $x \in X$ and $A \subseteq X$, by defining $d(x,A) = \inf\limits_{a\in A} d(x,a)$ if $A \neq \emptyset$, and $d(x,A) = 1$ otherwise. Further, we define *Hausdorff metric* for a pair $A, B \subseteq X$ as follows:

$$d_H(A,B) = \begin{cases} 0, & \text{if } A = B = \emptyset \\ \max\{\sup\limits_{a\in A} d(a,B), \sup\limits_{b\in B} d(b,A)\}, & \text{otherwise.} \end{cases}$$

The Hausdorff metric is one of the common ways of measuring resemblance between two sets in a metric space; it satisfies the conditions (M2) and (M3) in Definition 1, but it does not satisfy the condition (M1) in general.

III. Metric Controllability

Let us begin with the (finite) event set E of a DES G and a metric d on E which transfers E into a metric space. We now endow E^* with the *Baire metric* induced by d, which measures the distance between strings and pays more attention to the events occurring antecedently. Let $s = s_1 s_2 \cdots s_{l(s)}$ and $t = t_1 t_2 \cdots t_{l(t)}$ be two strings in E^*, and $l(s,t) = \max\{l(s), l(t)\}$. If $l(s) \neq l(t)$, say $l(s) < l(t)$, we take $s_i = \epsilon$ for each $i > l(s)$. We then define $\tilde{d}(s,t) = \sum_{i=1}^{l(s,t)} \frac{1}{2^i} d(s_i,t_i)$, where we set $d(\epsilon,\epsilon) = 0$, and $d(a,\epsilon) = d(\epsilon,a) = 1$ for any $a \in E$. It is easy to verify that $\tilde{d}$ does give rise to a metric on E^*. For later need, we make a useful observation.

Lemma 1: Let $L \subseteq L(G)$ and $s \in L(G)$. Then $\inf\limits_{t\in L} \tilde{d}(s,t) = \min\limits_{t\in L} \tilde{d}(s,t)$, namely, $\tilde{d}(s,L) = \min\limits_{t\in L} \tilde{d}(s,t)$.

As mentioned earlier, Hausdorff metric does not give rise to a metric space in general. However, if we consider the powerset $\mathcal{P}(E^*)$ of E^* with the Hausdorff metric induced by $\tilde{d}$, then we can get a metric space.

Proposition 1: Let $\tilde{d}_H$ be the Hausdorff metric induced by the metric $\tilde{d}$ introduced above. Then $(\mathcal{P}(E^*), \tilde{d}_H)$ is a metric space.

Proof: It follows directly from the definition of Hausdorff metric and Lemma 1. ■

The Hausdorff metric defined above measures the similarity of two languages. For convenience of notation, we will write d for the metrics $\tilde{d}$ and $\tilde{d}_H$ induced by d henceforth; it will be always clear from the context which metric is being considered. As a subset of E^*, $L(G)$ is a metric space with restricted metric. From now on, we will work in $L(G)$ instead of E^*, unless otherwise specified. We can now introduce the key notion.

Definition 2: Given $\lambda \in [0,1]$, a language $K \subseteq L(G)$ is said to be λ-*controllable* (with respect to $L(G)$ and E_{uc}) if there exists a language $\widetilde{K} \subseteq L(G)$ satisfying the following conditions:

1) $d(\widetilde{K}, K) \leq \lambda$;
2) $\widetilde{K}$ is controllable with respect to $L(G)$ and E_{uc}.

If such a $\widetilde{K}$ exists, we call it a *realization* of K.

Intuitively, a language K is λ-controllable if there is a controllable language that is similar to K. Observe that each controllable language is λ-controllable. The following example, however, shows that the converse is not true in general.

Example 1: Let $L(G) = \{\epsilon, a, ab, ag, af, abc, age\}$, $E_{uc} = \{c, g, f\}$, and $K = \{\epsilon, a, ab, af\}$. It is easy to see that K is not controllable. Let us now define a metric d on E as follows:

$$d(x,y) = \begin{cases} 0, & \text{if } x = y \\ 0.01, & \text{if } (x,y) = (b,g) \text{ or } (g,b) \\ 1, & \text{otherwise.} \end{cases}$$

Based on this metric, we can obtain the induced metrics on $L(G)$ and $\mathcal{P}(L(G))$, respectively. For example, $d(ab, ag) = 0.0025$ and $d(K, \{\epsilon, a, ag, af\}) = 0.0025$. Observe that $\{\epsilon, a, ag, af\}$ is controllable and it can serve as a realization of K whenever $\lambda \geq 0.0025$. Therefore, according to our definition, K is λ-controllable for any $\lambda \geq 0.0025$.

Let us give some remarks on the concept of λ-controllability.

Remark 1: 1) A language K is 0-controllable if and only if K is controllable. Note also that one can endow any event set E with discrete metric and educe further Hausdorff metric on $\mathcal{P}(E^*)$. Thus in view of this, the concept of λ-controllability is also a generalization of the ordinary controllability in the framework of Ramadge-Wonham.

2) If K is λ_1-controllable and $\lambda_1 \le \lambda_2$, then K is also λ_2-controllable. In particular, each controllable language is λ-controllable, for any $\lambda \in [0,1]$.

3) If K is λ-controllable, then so is $\overline{K}$. But the converse does not hold in general.

Specifying a metric on event set and determining the value of λ are two correlative and empirical issues which depend largely on control objective. They can be improved by stepwise refinements. The following are some useful properties of λ-controllable languages.

Proposition 2: 1) If K_1 and K_2 are λ-controllable, then so is $K_1 \cup K_2$.

2) If K_1 and K_2 are λ-controllable, then $K_1 \cap K_2$ need not be λ-controllable.

3) For any index set I, if each $\widetilde{K}_i$, $i \in I$, is a realization of K, then so is $\cup_{i\in I}\widetilde{K}_i$. But the intersection of two realizations is not necessarily a realization of K.

The following theorem presents a necessary and sufficient condition for a language $K \subseteq L(G)$ to be λ-controllable via its λ-ball in $L(G)$.

Theorem 1: A language $K \subseteq L(G)$ is λ-controllable if and only if $d(B(K,\lambda)^{\uparrow}, K) \le \lambda$.

Proof: The sufficiency follows immediately from the definition, so we only need to prove the necessity. Suppose that K is λ-controllable. By definition, there exists a controllable language $\widetilde{K} \subseteq L(G)$ with $d(\widetilde{K}, K) \le \lambda$. It follows that $d(x, K) \le \lambda$ for any $x \in \widetilde{K}$, that is, $\min_{y\in K} d(x,y) \le \lambda$ by Lemma 1. Therefore there exists $y_x \in K$ such that $d(x, y_x) \le \lambda$, and thus we get that $x \in B(y_x, \lambda) \subseteq B(K,\lambda)$ for any $x \in \widetilde{K}$. It means that $\widetilde{K} \subseteq B(K,\lambda)$, and furthermore, $\widetilde{K} \subseteq B(K,\lambda)^{\uparrow}$, which implies that $d(s, B(K,\lambda)^{\uparrow}) \le \lambda$ for any $s \in K$. Consequently, $\sup_{s\in K} d(s, B(K,\lambda)^{\uparrow}) \le \lambda$. On the other hand, since $B(K,\lambda)^{\uparrow} \subseteq B(K,\lambda)$, we have that $d(t,K) \le \lambda$ for any $t \in B(K,\lambda)^{\uparrow}$, which yields that $\sup_{t\in B(K,\lambda)^{\uparrow}} d(t,K) \le \lambda$. Hence, we obtain that $d(B(K,\lambda)^{\uparrow}, K) \le \lambda$. The proof is completed. ■

We would like to develop an algorithm for determining whether a finite language is λ-controllable. For this purpose, we need an algorithm for computing λ-ball about a string.

Let $G = (Q, E, \delta, q_0)$ be a deterministic automaton and $s = s_1 s_2 \cdots s_n$ be a fixed string in $L(G)$. Let d be the metric on $L(G)$ defined as before. For each $q \in Q$, define $E(q) = \{e \in E : \delta(q,e) \text{ is defined}\}$. Recall that $B(s,\lambda) = \{r \in L(G) : d(r,s) \le \lambda\}$.

Algorithm for $B(s,\lambda)$:

```
    $B(s,\lambda) \leftarrow \emptyset$;
    $r \leftarrow \epsilon$;
    $i \leftarrow 1$;
    $n \leftarrow l(s)$;
    $F(\lambda, r, i)$;
end Algorithm for $B(s, \lambda)$.
Here the procedure $F(\lambda, r, i)$ is defined recursively as follows:
*Procedure* $F(\lambda, r, i)$:
 if $i = n+1$ then
  if $\lambda \geq \frac{1}{2^n}$ then
     $B(s, \lambda) \leftarrow B(s, \lambda) \cup \{rr' \in L(G) : r' \in E^*\}$;
  else
     $k \leftarrow \lfloor -\log_2(1 - 2^n\lambda) \rfloor$;
     $B(s, \lambda) \leftarrow B(s, \lambda) \cup \{rr' \in L(G) : l(r') \leq k\}$;
  end if
  return;
 end if
 if $\lambda \geq \frac{1}{2^i} + \cdots + \frac{1}{2^n}$ then
  $B(s, \lambda) \leftarrow B(s, \lambda) \cup \{r\}$;
 end if
 for each $e \in E(\delta(q_0, r))$
  if $\lambda \geq \frac{d(s_i, e)}{2^i}$ then
   $r' \leftarrow re$;
   $\lambda' \leftarrow \lambda - \frac{d(s_i, e)}{2^i}$;
   $i' \leftarrow i + 1$;
   $F(\lambda', r', i')$;
  end if
 end for
end Procedure $F(\lambda, r, i)$.
```
■

The correctness of the above algorithm follows directly from the definition of Baire metric. The worst-case complexity of calculating $B(s, \lambda)$ is $O(|E|^{l(s)})$.

Based on the above algorithm and Theorem 1, we are now ready to provide an algorithm for determining whether a finite language is λ-controllable. Notice that $B(K, \lambda)^{\uparrow} \subseteq B(K, \lambda)$, so by definition we have that the condition $d(B(K, \lambda)^{\uparrow}, K) \leq \lambda$ in Theorem 1 holds if and only if

$\sup_{s\in K} d(s, B(K,\lambda)^{\uparrow}) \leq \lambda$. By Lemma 1, the latter is equivalent to that $\min_{b\in B(K,\lambda)^{\uparrow}} d(s,b) \leq \lambda$ for any $s \in K$. Further, this is equivalent to that $B(s,\lambda) \cap B(K,\lambda)^{\uparrow} \neq \emptyset$ for any $s \in K$. Note that $B(K,\lambda) = \cup_{s\in K} B(s,\lambda)$ and one can compute $B(K,\lambda)^{\uparrow}$ by using standard algorithm for the operation "$\uparrow$" developed in [22], [1], [7]. Summarily, we have the following result.

Theorem 2: To decide whether or not a finite language K is λ-controllable, we can follow the steps below:

1) For all $s \in K$, compute $B(s,\lambda)$ by using *Algorithm for* $B(s,\lambda)$.
2) Compute $B(K,\lambda)^{\uparrow}$ by using standard algorithm for the operation "$\uparrow$".
3) Decide whether or not each $s \in K$ satisfies that $B(s,\lambda) \cap B(K,\lambda)^{\uparrow} \neq \emptyset$.

If there exists $s \in K$ such that $B(s,\lambda) \cap B(K,\lambda)^{\uparrow} = \emptyset$, then K is not λ-controllable; otherwise, K is λ-controllable.

From 3) of Proposition 2, we see that the infinite unions of realizations of a λ-controllable language K are still a realization, which is the supremal realization of K. The next observation shows us the relationship between the supremal realization of K and the λ-ball about K.

Proposition 3: Let K be a λ-controllable language and $\widetilde{K}_i, i \in I$, be all realizations of K. Then $\cup_{i\in I} \widetilde{K}_i = B(K,\lambda)^{\uparrow}$.

We end this section with a discussion on supremal λ-controllable sublanguage. Let us define the class of λ-controllable sublanguages of K by $\mathscr{C}_{\lambda}(K) = \{M \subseteq K : M \text{ is } \lambda\text{-controllable}\}$. Observe that $\emptyset \in \mathscr{C}_{\lambda}(K)$, so the class is not empty. Define $K^{\Uparrow} = \cup_{M\in\mathscr{C}_{\lambda}(K)} M$. Note that 1) of Proposition 2 can be easily generalized to infinite unions, hence $K^{\Uparrow}$ gives rise to the largest λ-controllable sublanguage of K, where "largest" is in terms of set inclusion. We call $K^{\Uparrow}$ the *supremal λ-controllable sublanguage* of K and refer to "$\Uparrow$" as the operation of obtaining the supremal λ-controllable sublanguage. Clearly, $K^{\uparrow} \subseteq K^{\Uparrow}$. If K is λ-controllable, then $K^{\Uparrow} = K$. In the "worst" case, $K^{\Uparrow} = \emptyset$.

Several useful properties of the operation are presented in the following proposition.

Proposition 4: 1) If K is prefix closed, then so is $K^{\Uparrow}$.

2) If $K_1 \subseteq K_2$, then $K_1^{\Uparrow} \subseteq K_2^{\Uparrow}$. In other words, the operation $\Uparrow$ is monotone.

3) $(K_1 \cap K_2)^{\Uparrow} \subseteq K_1^{\Uparrow} \cap K_2^{\Uparrow}$; this inclusion can be strict.

4) $K_1^{\Uparrow} \cup K_2^{\Uparrow} \subseteq (K_1 \cup K_2)^{\Uparrow}$; this inclusion can be strict.

Recall that Proposition 3 tells us that $B(K,\lambda)^{\uparrow}$ is the supremal realization of λ-controllable language K. In fact, this result can be generalized to the case that K is not necessarily λ-

controllable.

Theorem 3: Let K be a language. Then $B(K,\lambda)^{\uparrow}$ is the supremal realization of $K^{\Uparrow}$.

Proof: By Proposition 3, we know that $B(K^{\Uparrow},\lambda)^{\uparrow}$ is the supremal realization of $K^{\Uparrow}$, so it is sufficient to show that $B(K,\lambda)^{\uparrow} = B(K^{\Uparrow},\lambda)^{\uparrow}$. Note that $K^{\Uparrow} \subseteq K$ and the operation $\uparrow$ is monotone, therefore we have that $B(K^{\Uparrow},\lambda) \subseteq B(K,\lambda)$, and furthermore, $B(K^{\Uparrow},\lambda)^{\uparrow} \subseteq B(K,\lambda)^{\uparrow}$. For the converse inclusion, set $K' = \{t \in K : \exists s \in B(K,\lambda)^{\uparrow}$ such that $d(s,t) \leq \lambda\}$, and then observe that $d(B(K,\lambda)^{\uparrow}, K') \leq \lambda$. Therefore, K' is λ-controllable, and $B(K,\lambda)^{\uparrow}$ is a realization of K'. Since $B(K',\lambda)^{\uparrow}$ is the supremal realization of K' by Proposition 3, we have that $B(K,\lambda)^{\uparrow} \subseteq B(K',\lambda)^{\uparrow}$. By the previous arguments, we know that $K' \in \mathscr{C}_{\lambda}(K)$. This means that $K' \subseteq K^{\Uparrow}$, and moreover, $B(K',\lambda)^{\uparrow} \subseteq B(K^{\Uparrow},\lambda)^{\uparrow}$. Consequently, $B(K,\lambda)^{\uparrow} \subseteq B(K^{\Uparrow},\lambda)^{\uparrow}$, as desired. ■

IV. OPTIMALITY OF REALIZATIONS

By introducing metric to the set of languages, we have defined the realization of a language K as the controllable language that is similar to K. Though there is an index λ reflecting the similarity, the elements of two similar languages may be quite different from each other. It is comprehensible since the similarity characterized by a metric yields that two elements are not identical unless the distance between them is 0.

In view of supervisory control, we are interested in finding a realization of K that has common elements with K as many as possible on the one hand and has different elements with K as few as possible on the other hand. To this end, let us consider the following problem.

Optimal Control Problem (OCP): Given $\lambda \in [0,1]$ and a nonempty language $K \subseteq L(G)$, find a supervisor S such that:

1) $d(L(S/G), K) \leq \lambda$.
2) $L(S/G)$ is *Pareto optimal* with respect to the following two sets which serve as measure of performance:
   - The *common element measure* of S, $CEM(S)$, defined as $CEM(S) = L(S/G) \cap K$.
   - The *different element measure* of S, $DEM(S)$, defined as $DEM(S) = L(S/G) \backslash K$.

Pareto optimality means that any possible improvement of $CEM(S)$ by enlarging this set is necessarily accompanied by an enlargement of $DEM(S)$. Similarly, any possible improvement of $DEM(S)$ by reducing this set is necessarily accompanied by a reduction of $CEM(S)$.

For simplicity, we suppose that the language K is prefix closed in this section. Let us first describe two extreme solutions to OCP.

Theorem 4: 1) OCP has a solution satisfying $CEM(S) = K$ if and only if $K^{\downarrow} \subseteq B(K, \lambda)$.

2) OCP has a solution satisfying $DEM(S) = \emptyset$ if and only if $d(K^{\uparrow}, K) \leq \lambda$.

The next theorem shows us that OCP has a solution whenever K is λ-controllable. It implies that we can obtain a Pareto optimal realization from any realization (in particular, the supremal realization). The resulting realization will significantly improve the original one.

Theorem 5: OCP has a solution if and only if K is λ-controllable.

The necessity of the above theorem is obvious. For the proof of the sufficiency, we need several lemmas. In fact, the process of proving the sufficiency is just the process of finding a Pareto optimal realization from a given realization.

Suppose that $\widetilde{K}$ is a realization of K, where $\widetilde{K}$ is prefix closed, but not necessarily Pareto optimal. We take two steps to find a Pareto optimal realization from $\widetilde{K}$: 1). Find a realization $\widetilde{K}_s$ by improving $\widetilde{K}$ such that $\widetilde{K}_s \cap K \supseteq \widetilde{K} \cap K$ and $\widetilde{K}_s \cap K$ is as large as possible, which helps us find more common elements; 2). Find a realization N_m by improving $\widetilde{K}_s$ such that $\widetilde{K}_s \cap K \subseteq N_m \subseteq \widetilde{K}_s$ and N_m is as small as possible, which helps us reduce the different elements.

For Step 1), define $\mathscr{M} = \{M : M \subseteq K \backslash \widetilde{K} \text{ and } M^{\downarrow} \subseteq K \cup \widetilde{K}\}$. Observe that $\emptyset \in \mathscr{M}$, so the class is not empty. Moreover, $\mathscr{M}$ is closed under arbitrary unions, hence it contains a unique supremal element, denoted M_s, with respect to set inclusion. Clearly, $M_s = \cup_{M \in \mathscr{M}} M$. The following lemma which is analogous to Lemma 5.1 in [3] provides some characterizations of M_s.

Lemma 2: 1) $M_s = M_s^{\downarrow} \cap (K \backslash \widetilde{K}) = (K \cup \widetilde{K})^{\uparrow} \cap (K \backslash \widetilde{K})$.

2) $\widetilde{K} \cup M_s^{\downarrow} = \widetilde{K} \cup M_s$.

The next proposition shows that by adding M_s to $\widetilde{K}$, we can get a better realization in the sense that the common elements may be improved without worsening the different elements.

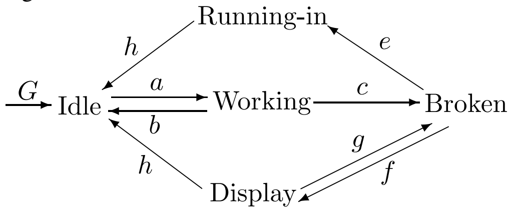
Proposition 5: Let $\widetilde{K}_s = \widetilde{K} \cup M_s$. Then $\widetilde{K}_s$ is a realization of K; moreover, $\widetilde{K}_s = \overline{\widetilde{K}_s}$, $\widetilde{K}_s \cap K \supseteq \widetilde{K} \cap K$, and $\widetilde{K}_s \backslash K = \widetilde{K} \backslash K$.

For Step 2), we require the following fact.

Lemma 3: Let $s \in E^*$, and suppose that there is a chain of prefix closed languages over E:

$$X_1 \supseteq X_2 \supseteq \cdots \supseteq X_i \supseteq \cdots$$

Fig. 1. Automaton G of Section V.



Tab. 1. Meaning of events.

Event	Event description
a	start (Controllable)
b	stop (Controllable)
c	fail (Uncontrollable)
e	replace (Controllable)
f	repair (Controllable)
g	reject (Uncontrollable)
h	approve (Uncontrollable)

satisfying that $d(s, X_i) \leq \lambda$ for all i. Then $d(s, \cap_i X_i) \leq \lambda$.

Let us now define $\mathscr{N} = \{N : \widetilde{K}_s \cap K \subseteq N \subseteq \widetilde{K}_s, N = \overline{N},$ and N is a realization of $K\}$. This class is not empty since $\widetilde{K}_s \in \mathscr{N}$ by Proposition 5. Recall that the intersection of two realizations of a λ-controllable language does not necessarily give a realization, so the class $\mathscr{N}$ has no infimal element in general. Nevertheless, we have the following result.

Lemma 4: The class $\mathscr{N}$ has a minimal element with respect to set inclusion.

Based on the previous lemmas, we can now prove the main result of this section.

Sketch of the Proof of Theorem 5: It remains to prove the sufficiency. Suppose that K is λ-controllable and $\widetilde{K}$ is a realization of K. Since K has been assumed to be prefix closed, it is easy to verify that $\overline{\widetilde{K}}$ is a realization of K. It follows from Lemma 4 that $\mathscr{N}$ has a minimal element, say, N_m. Further, one can prove by using Lemma 2 that N_m is a solution to OCP. ■

V. Illustrative Example

In this section, we apply the notion of λ-controllability to a machine which is a modified version of that studied in [6]. We explain the necessity of optimal control and illustrate the process of finding a Pareto optimal realization from an arbitrary realization.

The plant G, shown in Fig. 1, is a machine consisting of five states: *Idle, Working, Broken, Display,* and *Running-in*. The events of the plant model are listed in Tab. 1.

We suppose that the machine needs a thorough inspection after a period of run, say three "start" events for simplicity. Thus the control objective K is only concerned with strings that contain at most three a's. Furthermore, taking the effect of repair into account, K permits two occurrences of the "repair" event but only permits one occurrence of the "reject" event. More explicitly, K

Fig. 2. Automaton H of Section V.

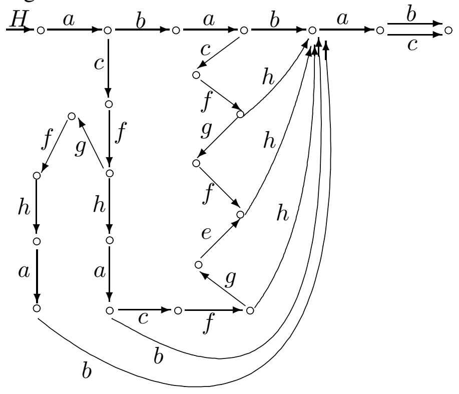
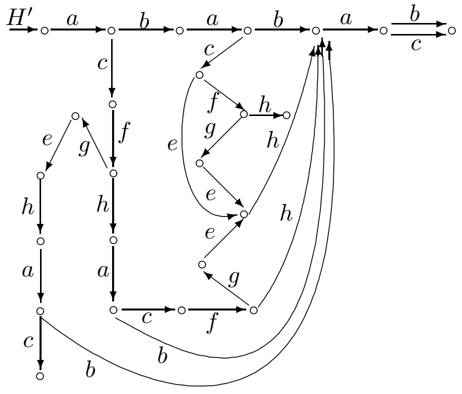


Fig. 3. Automaton H' of Section V.



is generated by the automaton H depicted in Fig. 2. Clearly, K is not controllable. Nevertheless, we can image that certain events such as “repair” and “replace” are similar, especially after some occurrences of “fail” and “reject”, since one would like to replace a component in some circumstances rather than repair it again and again. Notice that the machine needs an inspection after at most three occurrences of the “start” event and the length of the shortest path containing three a’s in H is 6, so we pay little attention to the difference between the “stop” event and the “fail” event after going through 6 occurrences of events. Experientially, we define a metric d on E as follows:

$$d(x,y)=\begin{cases}0, & \text{if } x=y\\ 2^{-7}, & \text{if } (x,y)\in\{(b,c),(c,b)\}\\ 2^{-10}, & \text{if } (x,y)\in\{(e,f),(f,e)\}\\ 1, & \text{otherwise}\end{cases}$$

and set $\lambda=2^{-14}$. We then find that K is 2^{-14}-controllable. In fact, it is not difficult to verify that the language $\widetilde{K}$ generated by the automaton H' depicted in Fig. 3 can serve as a realization. (Of course, there exist other realizations, for example, $B(K,\lambda)^{\uparrow}$.) On the other hand, we may still add some strings that belong to K and do not destroy the controllability of $\widetilde{K}$, and may also remove some strings in $\widetilde{K}$ but not in K and keep $\widetilde{K}$ controllable. Clearly, such a process of keeping K most possibly invariable is significative and necessary, and it can be accomplished by seeking a Pareto optimal realization as follows.

Keep the previous notation of the last section. By an easy calculation, we get that the supremal element M_s of $\mathscr{M}=\{M : M\subseteq K\backslash\widetilde{K}$ and $M^{\downarrow}\subseteq K\cup\widetilde{K}\}$ is $\{abacfha, abacfhab, abacfhac\}$, and thus $\widetilde{K}_s = \widetilde{K} \cup M_s$. Further, we have that

$$\begin{aligned}\mathscr{N} &= \{N : \widetilde{K}_s \cap K \subseteq N \subseteq \widetilde{K}_s, N = \overline{N}, \text{ and } N \text{ is a realization of } K\} \\ &= \{\widetilde{K}_s \backslash \{abace, abaceh, abaceha, abacehab, abacehac\}, \\ &\quad \widetilde{K}_s \backslash \{abaceha, abacehab, abacehac\}, \widetilde{K}_s \backslash \{abacehab\}, \widetilde{K}_s\}.\end{aligned}$$

Observe that $\mathscr{N}$ has one minimal element: $\widetilde{K}_s \backslash \{abace, abaceh, abaceha, abacehab, abacehac\}$, which gives rise to a Pareto optimal realization of K. It is worth noting that Baire metric plays a role here: although f is similar to e, f is not allowed to be replaced by e if the machine first breaks; in other words, any realization of K cannot contain the string ace.

VI. Conclusion and Discussion

In this paper, we have introduced a similarity-based supervisory control methodology for DES. By tolerating some similar behavior, we can realize some desired behavior which is uncontrollable in traditional SCT. A generalized notion of controllability, called λ-controllability, has been proposed. We have elaborated on some properties of λ-controllable languages and their realizations.

There are some limits and directions in which the present work can be extended. Note that the algorithm for $B(K, \lambda)$ developed in Section III only works for finite languages although all the remainder results have been established for any languages. It is desired to find a more general algorithm. Metrics chosen here including Baire metric and Hausdorff metric pay more attention to the events occurring antecedently. The distance between strings or languages that is given by such metrics may not be meaningful in certain practical systems, and the selection of these metrics is generally dependent on the particular problem considered. This means that perhaps supervisory control problems based on some other metrics or similarity measure (for example, Hamming distance in Information Theory) need to be considered. In addition, some other issues in standard SCT such as observability [12] and nonblocking [16] remain yet to be addressed in our framework.

Acknowledgment

The authors would like to thank Yuan Feng for some helpful discussions. Furthermore, the authors are very grateful to Associate Editor Alessandro Giua and the referees for their invaluable comments.

REFERENCES

[1] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham, “Formulas for calculating supremal controllable and normal sublanguages,” *Syst. Contr. Lett.*, vol. 15, pp. 111-117, Aug. 1990.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Norwell, MA: Kluwer, 1999.

[3] E. Chen and S. Lafortune, “Dealing with blocking in supervisory control of discrete-event systems,” *IEEE Trans. Automat. Contr.*, vol. 36, pp. 724-735, June 1991.

[4] J. W. de Bakker and E. P. de Vink, *Control Flow Semantics*. Foundations of Computing Series, Cambridge: MIT Press, 1996.

[5] S. Eilenberg, *Automata, Languages, and Machines: Volume A*. New York: Academic, 1974.

[6] R. Kumar, V. Garg, and S. I. Marcus, “Language stability and stabilizability of discrete event dynamical systems,” *SIAM J. Control Optim.*, vol. 31, pp. 1294-1320, Sept. 1993.

[7] S. Lafortune and E. Chen, “The infimal closed controllable superlanguage and its application in supervisory control,” *IEEE Trans. Automat. Contr.*, vol. 35, pp. 398-405, Apr. 1990.

[8] S. Lafortune and F. Lin, “On tolerable and desirable behaviors in supervisory control of discrete event systems,” in *Proc. IEEE Conf. Decision and Control*, Honolulu, HI, Dec. 1990, pp. 3434-3439.

[9] S. Lafortune and F. Lin, “On Tolerable and desirable behaviors in supervisory control of discrete event systems,” *Discrete Event Dynamic Syst.: Theory and Appl.*, vol. 1, pp. 61-92, 1991.

[10] Y. H. Li, F. Lin, and Z. H. Lin, “Supervisory control of probabilistic discrete-event systems with recovery,” *IEEE Trans. Automat. Contr.*, vol. 44, pp. 1971-1975, Oct. 1999.

[11] F. Lin, “Supervisory control of stochastic discrete event systems,” in *Book of Abstracts, SIAM Conf. Control 1990s*, San Francisco, 1990.

[12] F. Lin and W. M. Wonham, “On observability of discrete-event systems,” *Inform. Sci.*, vol. 44, pp. 173-198, 1988.

[13] T. Murata, “Petri nets: Properties, analysis and applications,” *Proc. IEEE*, vol. 77, pp. 541-580, Apr. 1989.

[14] C. A. Petri, *Interpretations of net theory*, St. Augustin: Gesellschaft fur Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-75-07, 2nd ed. Dez. 1976.

[15] P. J. Ramadge, “Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata,” *IEEE Trans. Automat. Contr.*, vol. 34, pp. 10-19, Jan. 1989.

[16] P. J. Ramadge and W. M. Wonham, “Supervisory control of a class of discrete event processes,” *SIAM J. Control Optim.*, vol. 25, pp. 206-230, Jan. 1987.

[17] P. J. Ramadge and W. M. Wonham, “The control of discrete event systems,” *Proc. IEEE*, vol. 77, pp. 81-98, Jan. 1989.

[18] A. Ray and S. Phoha, “Signed real measure of regular languages for discrete-event automata,” *Int. J. Control*, vol. 76, pp. 1800-1808, 2003.

[19] A. Ray, A. Surana, and S. Phoha, “A language measure for surpervisory control,” *Appl. Math. Lett.*, vol. 16, pp. 985-991, 2003.

[20] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. 2nd ed., Amsterdam; Boston: Academic, 2003.

[21] F. van Breugel, *Comparative Metric Semantics of Programming Languages: Nondeterminism and Recursion*. Boston: Birkhäuser, 1998.

[22] W. M. Wonham and P. J. Ramadge, “On the supremal controllable sublanguage of a given language,” *SIAM J. Control Optim.*, vol. 25, pp. 637-659, May 1987.